U.S. Patent Application of Alexander Sroka
for
Self-Organizing and Automatically Cross-Referencing Information Management System

Background -- Field of Invention

This invention relates to computer programs, specifically to a computer information management systems which organize and access information stored in computer systems.

Computer systems sometimes organize data into categories and items related to such categories. Categories may be organized in hierarchies or structures. For example, category "Companies" may contain a list of specific company names, like "Financial Services, Inc.", "Environment Care Corp." and "Computer Maintenance, Inc.". Items are specific textual strings containing a single word, two, three or up to couple hundred words text, or other information pieces.
Prior art does not handle efficiently practically unlimited number of categories and items and/or cannot do cross-referencing for such large number of categories and items.

Background -- Description of Prior Art

Currently, the prevailing implementations for Information Management Systems handling diversified pieces of information on the personal computers are utilized in Personal Information Managers (PIM). No single product on the market fulfills the requirements of easy interface, database strength, extendibility, flexibility or other specific features, like practically unlimited number of categories and items or, most importantly, automatic storage of cross-referencing between categories and items.

There are/were the following major competitors in existing PIM's market for personal computers:

- Maximizer ®-- with strong Btrieve database, no cross-referencing,
- Lotus Organizer® – strong visual interface, no database, no cross-referencing,
- Act! ® – database, contact manager, no cross-referencing,
- Lotus Agenda® – cross-referencing, unusable – DOS based, no database, now defunct,
- ECCO® – the best interface, no database,
- Lotus Notes® – not a PIM, but intelligent e-mail and document storing.

On the low end of information managing and scheduling programs are:

- InfoCentral® – information outlining,
- Schedule+® – networked scheduling,
- Network Scheduler® 3 – networked scheduling.

Lotus Agenda®, when it was available, is protected by the U.S. Pat. No. 5,115,504 issued to Belove et al. This patent is for a methodology which tries to accomplish a similar task, by a different system implemented in a DOS based database. The design of the system was cumbersome, limited to a linking file system and not utilizing the modern network data model design. It was creating more problems then it was solving.

Act! is recommended only for sale force automation, Maximizer is recommended for a broad spectrum of tasks, and Organizer is preferred by some users. Ease of interface is the single most important factor for the mass-market users. Then come the database and networking capabilities.

Summary Including Objects and Advantages

This invention is a specific software and/or hardware Database (Database) design and algorithms to access information in this Database. The Database is a model of the reality. The most prevalent characteristic of the Database is its **self-organizing** of information contained in the Database. The Database content is dynamic and effectively **changing the Database itself**. Humans always use **words** and phrases to describe the reality. The sentences are build primarily with **nouns** and **verbs**. The meaning of what is said depends on the **context** in which words are used, then it is concentrated on a particular **view** of the things and finally the things have their names. Database uses these words to classify the information. It is intuitive, but if somebody prefers to use a different naming convention, the Database can be adjusted by customizing it. The Database includes automatic cross-referencing algorithms to relate categories and items of information. Main design views of the Database are presented here -- others, not presented here variations of main design are covered also by this invention. The look and feel of the computer programs which deal with user friendly spreadsheet presentation of the Database information and particularly Name and Context combo are described. Any computer system with means of implementing the invention is also described.

References Cited

U.S. Patent Documents

5,115,504	5/19/1992	Belove et al.	395/600

Objects and Advantages

The object of the invention is to reduce redundancy of data storage, improve the performance of retrieving data and automate the process of categorizing information the way similar to human brain categorization.

Accordingly, several objects and advantages of the invention are:

1. A network database design, which easily categorizes and stores any number of pieces of information.

2. A database basic operation algorithms closely related to the database structure. The specific database design puts organization and structure to the way the specific pieces of information can be stored and browsed.

3. A database cross-referencing automatic algorithms are based on the database design, which stores any relationships between categories and items, categories and categories, items and items.

4. A database self-organizing and self-learning algorithms store statistical access or other information used to reorganize access paths to specific categories, items and their relationships. This information may be changed.

5. The look and feel of the user interface to the database. The interface uses a spreadsheet parallel to display in rows hierarchy of categories and items related to these categories. In spreadsheet columns are displayed categories which may be related to the items through other categories, which generally are sub-categories of the column categories.

6. Name and Context Combo in the user interface to the database. To accommodate display of different categories (Names) which may have the same Name, but a different meaning depending on the context they are used, the display includes means of viewing and manipulating Name/Context combinations.

7. A computer software program embodying a computer code for Personal Information Manager, said program comprising of subroutines for entering, viewing and editing of user's data with the spreadsheet interface, for retrieving of user's data with automatic cross-referencing of data and for system's self-learning based on statistical access information.

Still further objects and advantages will become apparent from a consideration of the ensuing description and accompanying drawings.

Concepts of Database

All drawings and preferred embodiment Data Definition Language use standard notation developed by the CODASYL Database Committee. The rectangular boxes represent Record Types, arrows represent Relationships between Records. Each arrow represents one to many relationship.

The main parts of the Database description are:

- Category Structure
- Categorizing Information
- Elements of Simplified Database
- Elements of Database
- Sample of the better developed Database
- Theory behind the Database
- Basic Cross-Referencing Algorithms of the Database
- Self Learning Algorithms of the Database

Category Structure

A database user builds his/her business by knowing his/her products and the procedures to deliver them. Database produces the Items of information (the product), by learning from user how user wants to find the Items (the procedure). The commonly used descriptors like date and time or the other preloaded Database categories do not need any explanation. But other categories may be at use only by a specific user's business, for example, manager's name or the names of companies a business is dealing with.
Database lets user put his/her knowledge into a structure, which is depicted in Figure 1. In a specific Context **1** user can define many Names **3**. The Relationship **2** between Context and Name is one to many optional on both sides of the Relationship **2**. For example, in Context Work a user can create Names of co-workers, Alex, Barbara, Jan and Agnes.

Each Name in the Database has a specific Context assigned to it at the time of creation of the Name in the Database.
A View **4** is a collection **5** of Names **6** (and their Contexts) as defined by the user at the time of the specific Name **6** entry. Context may also be used as a View.
Usually the View or Context **4** (or **7**) is used to have a collection of Names **6** (or **9**) in a required setup **5** (or **8**). Then, within the View or Context the category structure is like on Figure 2. Each Name in View can be used in a different Context.

For example, a View Activity can contain Names: Calls, Meetings, Mail and Follow up; a View People can contain Names: Peter, Jack, Barbara, Tiffany, Mark and Ken.
It is recommended, that a user starts working with the structure as shown on the Figure 2, without using Names from contexts other than Global. When user sees the need for differentiating the meanings of a particular word in different contexts, then he/she can start to use different contexts. For example, there is already created a View People with Names: Peter, Jack, Barbara, Tiffany, Mark and Ken, containing names of the people from the business. Then user entered an Item with the text "Call Barbara at 2:30 PM", meaning to call user's wife, which has the first name Barbara, same as one of user's coworkers. The system has assigned the Item "Call Barbara at 2:30" to Barbara from the business, because it didn't know about user's wife. Now, what user is supposed to do, is to add Barbara from View People, to a new Context Home. In effect, Barbara will be related to two Contexts, namely Work and Home.

Categorizing Information

User enters the pieces information to the system through Items **11 (109)**. Items are entered on the View Form (screen) as in Figure 18. User enters the Items **11 (109)** for a Name **9 (106, 108)**. Each Item may be assigned **10** to many Names **(106, 112)**. The full structure of Database system is shown on Figure 3.

User can create as many as he/she wishes Contexts, Views, to each View can be assigned many Names, and to each Name can be assigned many Items. Each Item may be assigned to many Names. **There is no limit on how many of these elements can be created.**

Contexts, **Views** and **Names** together create **Categories 12**. All Categories **12**, all Items **14** and all many to many connections **13** between Categories and Items in Database are stored in Database called **Neuron**. The simplified structure of Database is depicted as shown in Figure 4.

The presented in Figure 3 and in Figure 4 structure of Database is totally flexible and extendible, because there can be created many levels of classification. For example, a user can have the structure of categories and an Item, as shown on Figure 5. In this Figure 5 arrows **16, 18, 20, 22, 23, 27, 28** and **29** depict the physical pointers between specific categories/items. These pointers are merely one of possible physical implementation of relationships.

In this example user has five categories: Activity **15**, Follow Up **17**, Leads **19**, NOVACORP **21**, Smith **22'** and an Item "Call Smith tomorrow and meet him Monday" **30**. Each of the categories can become a View and/or a Name. At the time of the entry the Item "Call Smith tomorrow and meet him Monday" is automatically assigned to other categories: Calls **24**, Tomorrow **26**, Meetings **25**; and other standard (not shown here) date categories: Tomorrow's Date, Monday (next) and next Monday's Date. All categories can have their own structures or can participate in any structures. Together, all category structures in the Database reflect user's own information network.

Elements of Simplified Database

In the previous section **Categories 31** and **Items 36** were described. They are the most frequently used **elements** of Database. The **Notes 33** are the next element of Database. Notes are not utilized in the automatic assignments, like Items and Categories, but are useful to store the additional information attached to any Category or Item. As the Items are limited in size, to make their processing efficient, the Notes are actually unlimited in size. One note can have many pages, and one page contains around one printed page of text in the preferred embodiment. The number of notes is practically unlimited. Notes are attached **32, 35** to Categories and Items, but there can be also notes by themselves not connected to any other element of Database. However, it is recommended that Notes are attached to at least one Category or Item, because this way they can be easily found out. Otherwise, a user has to remember note's identifier or would have to scan all notes, to find the right one. Figure 6 shows how the Notes relate to other elements of Database.

The Simplified Database is made up of Items, Categories and Notes. All these database elements can be related to each other **32, 34, 35** in many to many relationships.

Elements of Database

The basic configuration of the Database allows to store any information. Categories are divided into Nouns and Verbs. A part of the Database is called the **Noun 37** branch, because all parts in this part are nouns. Similar structure is created for the verbs. This part of the Database is called the **Verb 38** branch. The full Database contains both branches being connected by **Item 41**. Use of one or both branches constitutes use of the Database. Figure 7 shows how all three elements relate to each other in the Database. Additional long pieces of text are stored in **Notes 42**. The many to many relationships **37"**, **39**, **40**, **37'** and **38'** relate Elements of Database to each other.

Sample of the better developed Database

In any database developed on the four element Database as shown in Figure 7, the branches get more complicated to store internal to the branches relationships and additionally relationships between the branches.

Full implementation of the invention should create two databases:
- one without Items – called Dictionary and
- one with Items – called Reality.

Figure 8 shows one of possible implementations of the Database developed with Items. The database has the Start **43** record, which is used as the owner of the sets **44** and **62** for sequential retrieval of Main Noun **45** and Main Verb **63** records.
The Noun Branch is developed to classify and store the hierarchical and network relationships between the elements of the Noun Branch. Main Noun record **45** is the owner of a collection **45'** of Noun Group Records **46**. Noun Group **46** is the owner of a collection **45"** of Noun Records **49**. Structure Record **47** is the record to store the many to many relationships **50** between Noun Group Records **46**. These relationships **50** are implemented as a double set from Noun Group Records **46** to Structure Record **47**. Structure Record **48** is the record to store the many to many relationships **51** between Noun Records **49**. These relationships **51** are implemented as a double set from Noun Records **49** to Structure Record **48**. Item Record **55** is related many to many to the Noun Record **49** via Noun-Item Relationship Record **53** and relations **54** and **54'**.
The Verb Branch is developed to classify and store the hierarchical and network relationships between the elements of the Verb Branch. Main Verb record **63** is the owner of a collection **63'** of Verb Group Records **64**. Verb Group **64** is the owner of a collection **63"** of Verb Records **65**. Structure Record **66** is the record to store the many to many relationships **68** between Verb Group Records **64**. These relationships **68** are implemented as a double set from Verb Group Records **64** to Structure Record **66**. Structure Record **67** is the record to store the many to many relationships **69** between Verb Records **65**. These relationships **69** are implemented as a double set from Verb Records **65** to Structure Record **67**. Item Record **55** is related many to many to the Verb Record **65** via Verb-Item Relationship Record **72** and relations **73** and **73'**.
Structure Record **75** is the record to store the many to many relationships **74** between Item Records **55**. These relationships **74** are implemented as a double set from Item Records **55** to Structure Record **75**.
Noun Record **49** is related many to many to the Verb Record **65** via Noun-Verb Relationship Record **71** and relations **52** and **70**.
Item **55** is further analyzed into Item Analysis Records. This schema has Item Analysis Records **57**, **58**, **59** and **60**. Item Analysis Records are related to Item Record **55** via relation(s) **56**. These relations may be implemented as a single set or multiple sets.
Note Record **61** in this schema is not related directly to Noun **49**, Verb **65** or Item **55**. There are direct relations, but for efficiency reasons they are implemented as direct Noun **49**, Verb **65** or Item **55** key duplication in Note Record **61**.

Theory behind the Database

The Database follows theoretically the analysis of sentences in any language. Full sentence in any language contains Nouns, Verbs and quantified information about the noun operated by the verb. Such quantified information can be fully analyzed. Some sentences are not fully quantified, but logically they still follow the full model. (Parts are less than the whole).
By definition, elementary information is a simple sentence describing state of observed event or process.
Quantified elementary information, by definition, is a sentence with an argument describing result of measuring of an object state.
In the most complicated form a quantified elementary information contains results with discrete measurement result.

Name { = <= >= < > } Number

(any of)

Where:
Name – name for the Number
Number – Real Number
=, <=, >=, <, > – functors creating sentences, semantics like in theory of real numbers

Quantified elementary information is separated into three parts:

- Noun,

- Verb,

- Item.

Noun and Verb are analysis of the Name, Item contains the Number.

Using network data model notation (CODASYL) the model in Figure 9 directly translates quantified elementary information into a database language. Noun **76** is in many to many relationship **77** to Verb **78**. Verb **78** is in many to many relationship **79** to Item **80**.
This global data model is followed in all databases for quantified elementary information.

However, the described above analysis has mainly theoretical significance. In practice, it can be used to estimate the Database size.

Practically, the following schema should be utilized, see Figure 10. It is based mostly on the type of query directed towards the database. Also, in reality, multiple Nouns in quantified elementary information databases are analyzed in same Verbs -- so Verbs are becoming independent of Nouns. Noun **81** is in many to many relationship **83** to Item **85**. Verb **82** is in many to many relationship **84** to Item **85**.

The Database is made of Nouns, Verbs and Items. What is critical in this diagram, are the relationships between the basic elements of the sentence. Names given to the basic elements of the

sentence are here only for clarity of definition. The essence of this design are the relationships between the parts of the analysis, not the specific names for them.

Based on the aforementioned, in Figure 10, schema using network model notation all systems will be build. In reality all the elements of the diagram become developed into finer analysis of them, to accommodate higher speed of retrieval within databases and to simplify the navigation in specific databases for a specific area of knowledge being analyzed.

To accomplish conversion from network model to relational model of databases, two relations are being defined:

1. Relation R1: N R1 V – in Noun N exists Verb V

2. Relation R2: V R2 I – in Verb V exists Item I

These two relations must coexist on the same set of Verbs, which is closed on the Product operation.

To have a properly build database, two thesis have to be true:

1. Each Item in the database has to be assigned to a Verb or combination of Verbs.

2. Each Verb has to be assigned to a Noun or combination of Nouns.

Simply speaking, there are no Items with nonexistent Verbs and/or Nouns.
The reverse functions also cooperate.

Basic Cross-Referencing Algorithms of the Database

The basic algorithms of the Database traverse the fully developed Noun and/or Verb branches to access the Items. Items by themselves can be accessed in a regular sort (index) sequence. To utilize the power of the Database the retrieval queries have to limit the number of retrieved Items based on the query parameters.
The sample of the Noun Branch is depicted in Figure 11. The basic query reads a Noun **86** as the specified View, takes it as the head of the Structure list **87** and scans all the elements belonging to the list using the Structure record using the connecting sets **86"**. The elements of the View list create the headings **92** of the result spreadsheet in Figure 12. Then, for a specified Name **94** with Context **94"** treated as the head of another list it reads all the elements belonging to the Name list using the Structure record **87** and the connecting sets **86"**. Then, the connecting sets **86'** and **89'** and Noun-Item Relationship Record **88** is used to find out all Items **89** (**93**) belonging to the Name list elements; all the Nouns **86** for these Items are read and if any of the these Nouns appear on the list which head is specified in the heading **92**, than such Noun **86** (**91**) is printed in the intersection of the Item **89** (**93**) and the heading **92**.

Same algorithm may be used to traverse the Verb Branch.
If the Database schema is developed as in Figure 8, then the basic algorithm can be extended to utilize the classifications of Nouns (Verbs), which are the records **45**, **46** (**63**, **64**) above the main Noun (Verb) records **49** (**65**).

Illustration of result of the queries is the screen printout in Figure 12. The expected result of the query is a logical intersection of View **90** with Name **94**. Each returned Item **93** has to belong to sets: one is the View set **90** or its sub-name **92** and another is the Name set **94** or its sub-name **94'**. This way the user is presented only with Items **93** that fulfill this two dimensional query parameters.

Self Learning Algorithms of the Database

Further referring to Figure 11, the lists **86"**, **86'**, **89'** mentioned in the previous section for the Basic Cross-Referencing Algorithms of the Database are ordered by key value in records Structure **87** and Noun-Item Relationship Record **88**. Same applies to all Structure and Relationship Records in all Figures. Every time the specific link between the lists is utilized, the key value is incremented by a discrete count. This count can be also inputted by user on request. This organizes the list and strengthens the connection between the list head and its element. The next time the same list gets read, the higher count, stronger elements are read first. This constitutes the self-learning process of the Database.
Basic structure of Noun (also Verb) record is presented in the Figure 13. It contains Context Code **95** and Noun (Verb) Name Value **96**. For example, Context Code 0 (zero) for Global Context and Name with value 'New York'.

Figure 14 shows the minimal content of the Relationship (or Structure) record containing usage count or certainty factor **97**.

The Noun (Verb) data is a Product of any number of multiple other Nouns (Verbs) and their Relationship usage counts **100**, like in Figure 15. Nouns (Verbs) each carry Context Code **101** and Noun (Verb) Name Value **102**.
Example given in Figure 16 contains Noun data, which is built from four other products of Noun Name Values **105** and their Contexts **104**; and their relationship count **103**.

Name and Context Combo and Spreadsheet Interface to the Database

Referring to Figure 18, visual representation of the Database content is provided. Each Name in the Database is stored not only as a value of its content, but also in conjunction with its Context. To let a user view and edit the content of such pair of objects, the Name and Context Combo is developed. In the preferred embodiment, the only existing implementation of the invention such Combo is represented by means of two closely visually related list boxes: List Box Name **106** and List Box Context **107**. In other screens of the preferred embodiment Name and Context Combo can be shown as Name and Context columns of a screen. Other visualizations are may display the same content in a different layout.

Referring to Figure 18, visual representation of the Database content is also provided for multiple hierarchical and other relationships between Categories and Items. Each Name from the Database may have a list of sub-Names **108** (displayed here with the folder picture). These sub-Names may again be related to their sub-Names. Each Name or sub-Name may have Items **109** (displayed here with the page picture) related to them and displayed in proximity of such Name or sub-Name. The list of columns **110** is the list of Names to which a user wants to view relationships to Items. Such list of columns is named as View and such Name **111** is displayed in the View List Box **111**. The Name(s) **112** displayed in the cell related to a row and column are the Name(s) relating the name(s) in the column heading **110** to a Name(s) or Item(s) in the related row **109** or **108**.

For example, referring to Figure 18, Name "jackie" from Context "global" has Items "call alex soon", "see jackie", another "see jackie" and "call alex mondaa". Name "jackie" from Context "global" has also sub-Name "people". View "manager" has sub-Names (and columns) "contact", "people", "completed work1", "assigned date" and "alarm". Item "call alex soon" is related to column "people" through Name "alex", which is displayed in the intersection of the row for the Item and column for Name "people". Item "see jackie" is related to column "people" through Name "jackie", which is displayed in the intersection of the row for the Item and column for Name "people". Another Item "see jackie" is related to column "people" through Name "jackie", which is displayed in the intersection of the row for the Item and column for Name "people". Item "call alex mondaa" is related to column "alarm" through Name "call alex mondaa", which is displayed in the intersection of the row for the Item and column for Name "alarm".

Brief Description of the Drawings

Figure 1. illustrates Category Structure dependencies for Context and Names.
Figure 2. shows Standard Category Structure for the View Context.
Figure 3. shows Structure of the Database Information Model.
Figure 4. is the diagram of the Simplified Structure of the Database Information Model.
Figure 5. shows structure of a sample category classification.
Figure 6. shows all Elements of the Simplified Database Information Model.
Figure 7. shows all Elements of the Database.
Figure 8. displays realistic schema of the Database for Reality.
Figure 9. is a diagrammatical representation of quantified elementary information.
Figure 10. is a diagrammatical representation of all elements of the quantified elementary information.
Figure 11. contains illustration for the basic retrieval algorithm.
Figure 12. displays two dimensional query results of the basic retrieval algorithm.
Figure 13. shows basic structure of Noun (Verb) record.
Figure 14. shows basic structure of Relationship (or Structure) record containing usage count or certainty factor.
Figure 15. displays product of Nouns (Verbs) and Structure of Nouns (Verbs); with three elements.
Figure 16. shows example of product of Nouns (Verbs) and Structure of Nouns (Verbs); with four elements.
Figure 17. contains the RAIMA® Database Data Definition Language Schema for BrainAgenda©.
Figure 18. displays two dimensional query results of the basic retrieval algorithm and shows spreadsheet interface. The Name and Context Combo is displayed.

Summary Including Objects and Advantages

This invention is a specific software and/or hardware Database (Database) design and algorithms to access information in this Database. The Database is a model of the reality. The most prevalent characteristic of the Database is its **self-organizing** of information contained in the Database. The Database content is dynamic and effectively **changing the Database itself**. Humans always use **words** and phrases to describe the reality. The sentences are build primarily with **nouns** and **verbs**. The meaning of what is said depends on the **context** in which words are used, then it is concentrated on a particular **view** of the things and finally the things have their names. Database uses these words to classify the information. It is intuitive, but if somebody prefers to use a different naming convention, the Database can be adjusted by customizing it. The Database includes automatic cross-referencing algorithms to relate categories and items of information. Main design views of the Database are presented here -- others, not presented here variations of main design are covered also by this invention. The look and feel of the computer programs which deal with user friendly presentation of the Database information and particularly Name and Context Combo are described. A computer system with means of implementing the invention is also described.

Preferred Embodiment – Detailed Description

Detailed descriptions of the preferred embodiment are provided herein. It is to be understood, however, that the present invention may be embodied in various forms. Therefore, specific details disclosed herein are not to be interpreted as limiting, but rather as a basis for the claims and as a representative basis for teaching one skilled in the art to employ the present invention in virtually any appropriately detailed system, structure or manner.

The preferred embodiment is to be done using a Microsoft Windows® compatible computer. On such computer a Network Model Database Manager software has to be installed. Database definition for a Network Model Database Manager is stored in the Data Definition Language format file. The only existing embodiment of the invention is in BrainAgenda© Personal Information Manager software. The Data Definition Language format file is created using RAIMA® Network Model Database Manager. Figure 17 show the Data Definition Language format file specific to and working with RAIMA® Network Model Database Manager. The Data Definition Language format file stores the database definition, which directly relates to some claims of the invention. This file is called a Schema of the Database.

Computer Database Manager Listing for the Database Design is represented in Figure 17. This is a Network Model Database design using a specific RAIMA® Database Manager. The Database Design may be directly implemented in a specific computer by supplying the Listing to the RAIMA® Database Manager. This schema is the implementation of realistic schema of the Database for Reality as shown in Figure 8. Additional elements included in the schema and not represented in the Figure 8 are utilized mostly for performance improvement.

In Figure 17 all lines and each string starting with two characters / (forward slash) and * (asterisk) and ending with * (asterisk) and / (forward slash) are the comment lines or strings. Comment string contains text which helps a database analyst understand database features. Comment text is irrelevant to the database manager interpretation of the database definition.
In this description the references are done by the name of the element as it is used in the Figure 17. The **database BRAIN** is defined with the page size of **6144** bytes of storage. The file **F100010.00** contains records of type **noun**. The file **F100011.00** contains records of type **datar** and **datar_tabl**. The file **F100012.00** contains records of type **noun_datar**, **noun_str**, **noun_synonim**, **datar_str**, **action_before** and **action_after**. The file **F100019.00** contains records of type **brain** and **note**. The key file **F100010.00K** contains keys of type **noun.id**. The key file **F100011.00K** contains keys of type **datar.id**. The key file **F100019.00K** contains keys of type **note.id**.
Field types are defined as per language C conventions as char (character) with length in brackets, long (numerical) and double (numerical, with double size). The `struct` keyword is used to designate the structure name, which includes multiple fields. The structure name may be used in programming language (for example, C) to manipulate the whole named group of fields instead of single fields. The structure name does not change the way the included fields behave.

Definition for record type **brain** contains multiple fields `db_path, db_name, type_v, kname_v, subtype_v, name_v, type_n, kname_n, subtype_n, type2_n, kname2_n, subtype2_n, name_n, read_action, next_1, next_2, next_3, value_1, value_2, value_3, double_1, double_2, double_3, reserve_1, reserve_2, free`, which are not specifically related to the invention. These fields are defined for ease of programming to store some additional information related to the database as a whole.
Definition for record type **brain** contains also groups of fields `id_v` and `id_n`.

Definition for record type **noun** contains multiple fields `type, kname, subtype, type2, kname2, subtype2, name, type_p, kname_p, subtype_p, cf, delete, joint_id, read_action, date_create, date_when, date_done, date_start, date_end, short_name, cat_type, exclusive, settings, layout_link, type_link, kname_link, subtype_link, type_note, kname_note, subtype_note, position_note, free_1, free_2, reserve_1, reserve_2, reserve_3.`
Definition for record type **noun** contains also groups of fields `id, id_p, id_link` and `id_note`. Group of fields `id` relates to basic structure of Noun (Verb) as shown in Figure 13. CONTEXT Code **95** relates to field `type`, NOUN Name Value **96** relates to field `kname`. Field `kname` contains first 40 bytes of the full NOUN Name Value **96**. Field `name` contains all 255 bytes of the NOUN Name Value **96**. In the Preferred Embodiment is implemented product of Nouns (Verbs) and Structure of Nouns (Verbs) with two elements as related to Figure 15. As related to Figure 15, CONTEXT Code 1 **101** relates to field `type`, NOUN Name Value 1 **102** relates to field `kname`. Field `kname` contains first 40 bytes of the full NOUN Name Value 1 **102**. CONTEXT Code 2 **101** relates to field `type2`, NOUN Name Value 2 **102** relates to field `kname2`. Field `kname2` contains first 40 bytes of the full NOUN Name Value 2 **102**. Definition for record type **noun** relates directly to Noun **49**.

Definition for record type **datar** contains multiple fields `type, kname, subtype, name, cf, delete, joint_id, read_action, date_create, date_when, date_done, date_start, date_end, settings, type_note, kname_note, subtype_note, position_note, long_1, reserve_1, reserve_2, reserve_3, reserve_4.`
Definition for record type **noun** contains also groups of fields `id` and `id_note`. Field `kname` contains first 40 bytes of the full Item **55**. Field `name` contains all 255 bytes of the Item **55**. Definition for record type **datar** relates directly to Item **55**.

Definition for record type **datar_tabl** contains multiple fields `elem, cf, delete, date_create, read_action, double_1, reserve_1, reserve_2.`

Definition for record type **note** contains multiple fields `from, type, kname, subtype, page_nr, name, cf, chapter, chapter_1, chapter_2, chapter_3, chapter_4, chapter_5, chapter_6, verse, page, delete, read_action, reserve_1, reserve_2, reserve_3, reserve_4.`
Definition for record type **note** contains also group of fields `id`. Field `kname` contains first 40 bytes of the full `page`. Field `page` contains all 5000 characters of a page of text stored in the database. .
Definition for record type **note** relates directly to Note **61**.

Definition for record types **noun_str, noun_datar, datar_str** contains multiple fields `cf, date_create, read_action, double_1, reserve_2, reserve_3`. Field `cf` relates to basic structure of Relationship (or Structure) record containing usage count or certainty factor as shown in Figure 14 Count **97**. In the Preferred Embodiment is implemented product of Nouns (Verbs) and Structure of Nouns (Verbs) with elements as related to Figure 15. As related to Figure 15, Count 1 or Count 2 or Count 3 **100** relates to field `cf`. Definition for record types **noun_str, noun_datar, datar_str** relates directly to Structure records **48, 53, 75**.

Definition for record types **action_before, action_after, noun_synonim** contains multiple fields `cf, date_create, read_action, double_1, reserve_2, reserve_3.`

All sets in the schema are ordered descending by the `cf` fields from the member records.
Definition for set **noun_set** contains record **brain** as the owner and record **noun** as the member.
Definition for set **datar_set** contains record **noun** as the owner and record **noun_datar** as the member.
Definition for set **datar_noun_set** contains record **datar** as the owner and record **noun_datar** as the member.

Definition for set **noun_synonim_exp_set** contains record **noun** as the owner and record **noun_synonim** as the member.
Definition for set **noun_synonim_imp_set** contains record **noun** as the owner and record **noun_synonim** as the member.
Definition for set **noun_exp_set** contains record **noun** as the owner and record **noun_str** as the member.
Definition for set **noun_imp_set** contains record **noun** as the owner and record **noun_str** as the member.
Definition for set **datar_exp_set** contains record **datar** as the owner and record **datar_str** as the member.
Definition for set **datar_imp_set** contains record **datar** as the owner and record **datar_str** as the member.
Definition for set **action_before_exp_set** contains record **noun** as the owner and record **action_before** as the member.
Definition for set **action_before_imp_set** contains record **noun** as the owner and record **action_before** as the member.
Definition for set **action_after_exp_set** contains record **noun** as the owner and record **action_after** as the member.
Definition for set **action_after_imp_set** contains record **noun** as the owner and record **action_after** as the member.
Definition for set **datar_tabl_set** contains record **datar** as the owner and record **datar_tabl** as the member.

While the invention has been described in connection with a preferred embodiment, it is not intended to limit the scope of the invention to the particular form set forth, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims.

Preferred Embodiment -- Operation

Algorithms of the Database described in this invention are implemented in BrainAgenda©. Specific source code of programs which perform the algorithms is written for the current usage of the Database in the Personal Information Manager operation.

The basic algorithms of the Database traverse the fully developed Noun and/or Verb branches to access the Items. Items themselves can be accessed only in a regular sort (Index) sequence. To utilize the power of the Database the retrieval queries have to limit the number of retrieved Items based on the query parameters. Good illustration of result of the queries is the screen print in Figure 12. The expected result of the query is a logical intersection of View (and Context) with Names (and their Contexts). Each returned Item has to belong to two sets: one is the View set and another is the Name set. This way the user is presented only with Items that fulfill said two dimensional query parameters.
Referring to Figures 8, 12 and 18, the basic query reads a Noun Record (**49**, **noun**) as the specified View (**111**), takes it as the head of the list and scans all the elements belonging to the list using the STRUCTURE (**48**, **noun_str**) record. The elements of the View list create the headings (**110**) of the result spreadsheet. Then, for the specified Name (and Context) reads a Noun Record (**49**, **noun**) treated as the head of another list it reads all the elements (**108**) belonging to the Name list using the

STRUCTURE (**48**, `noun_str`) record. Then, for all the Items (**55**, `datar, 109`) belonging to the Name list elements all the Nouns (**49**, `noun, 112`) are read and if any of the these Nouns appear on the list which head is specified in the heading (**110**), than such Noun is printed in the intersection (**112**) of the Item and the heading.
Same algorithm may be used to traverse the Verb branch. The basic algorithm can be extended to utilize the classifications of Nouns (Verbs) -- the records above the main Noun (Verb) records.

Other Embodiments

Intelligent Device -- Description

Any intelligent Device has to have Knowledge Bank. This Knowledge Bank may be implemented in the computer by utilizing database as described in this invention.

Intelligent Device -- Operation

Any intelligent Device has to have Knowledge Bank. This Knowledge Bank may be implemented in the computer by utilizing database operation algorithms as described in this invention.

Conclusions, Ramifications, and Scope

Accordingly, it can be seen that this database design allows a system, which uses the invention, to perform highly effective storage of any number of pieces of information and their relationships to other pieces of information. As such the invention may be, and is, utilized for storage of dissimilar pieces of information in practical implementations especially useful for Information Managers and Knowledge Banks.

Although the description of the invention embodiment contains many specificities, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the presently preferred embodiments of this invention. Various other embodiments and ramifications are possible within invention's scope. For example, this database design allows a system, which uses the invention to perform highly effective storage of any language sentences (languages different than English). Above that, it also performs functions similar to human brain, namely, it stores unlimited number of connections between pieces of information, automatically cross-references them and self-organizes them according to any learning pattern, for example, frequency of usage of a piece of information in relationship to other pieces of information.

Any computer system with means of implementing the invention is said to be containing the invention. Thus the scope of the invention should be determined by the appended claims and their legal equivalents, rather than by the examples given.